REMARKS/ARGUMENTS

These remarks are made in response to the Office Action of July 2, 2004 (Office Action). As this response is timely filed within the 3-month shortened statutory period, no fee is believed due.

In paragraphs 2 and 3 of the Office Action, claims 1-2, 10, 16-17, 24-25, 33, and 39-40 and 47 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent Number 6,363,421 to Barker *et al.* (Barker) in view of U.S. Patent Publication No. 2002/0169878 to Orenshteyn (Orenshteyn). In paragraph 4 of the Office Action, claims 5-9, 11, 13-15, 20-22, 28-32, 34, 36-38, and 43-45 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over Barker in view of Orenshteyn and in further view of Sun Microsystems: JAVA2 Management Extensions White Paper, Dynamic Management for the Service Age (JAVA2). In paragraph 5, claims 12, 23, 35, and 46 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over under Barker in view of Orenshteyn in further view of JAVA2, in further view of U.S. Patent Number 6,633,923 to Kukura *et al.* (Kukura).

Prior to addressing the rejections on the art, a brief review of the Applicants' invention is in order. Applicants disclose a system, method, and apparatus for managing resources of an application, where the application runs within a distributed environment having application components residing within different software machines, each of which can reside upon a different host computer. Each software machine can be a virtual machine, such as a Java Virtual Machine. Distributed application components can utilize remotely located resources residing within one or more software machines. Communications between the application components and the software machines can be utilize a centralized communication intermediary.

BEST AVAILABLE COPY

More specifically, the disclosed system includes a master agent that can receive management commands from several different application components. The master agent can direct received commands to mini-agents that are each coupled to one or more application resources. Each mini-agent can handle local overhead for the application resources, triggering the utilization of these localized resources responsive to the management commands received from the master agent. Moreover, each mini-agent can be communicatively linked to and registered with the master agent. The master agent can communicate with each mini-agent using standardized commands, which are translated by the mini-agents into local commands specific to the local application resources managed by each mini-agent.

By decoupling application components (software routines that consume resources) from application resources consumed during program execution (manageable resources locally managed by the mini-agents), applications can be easily be deployed in a distributed computing space. Accordingly, the master agent can function as a layer of abstraction between application components consuming resources and the resources consumed. Each mini-agent can translate standard commands issued by the master agent to a local environment.

Turning specifically to the rejections on the art, in paragraphs 2 and 3 of the Office Action, claims 1-2, 10, 16-17, 24-25, 33, and 39-40 and 47 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over Barker in view of Orenshteyn.

Barker is not directed towards the problems with deploying **applications** within a distributed computing space. Instead, Barker teaches a Web-based administrative tool for remotely managing network hardware associated with a network server. In Barker, the network hardware is referred to as a network element 14 defined as "(a device) responsible for processing

event and alarm notifications to the element management system via SNMP ..." as stated at column 4, lines 56-59. Barker teaches that an Web browser can be used as the remote Web-based administrative tool in lieu of an administrative interface local to the network server. Contemplated interfaces for the Web browser are shown in FIGS. 10-13.

One skilled in the art recognizes that SNMP refers to the Simple Network Management Protocol (SNMP) that is a network management protocol used in TCP/IP networks. SNMP is used to manage network devices, such as hubs, routers, bridges, etc. In SNMP, data is passed from SNMP agents that report activity of associated network devices to a workstation console that oversees the network. The workstation console relies upon a Management Information Base (MIB), which is a data structure that defines what is obtainable from the network device and what device operations can be controlled (turned off, on, etc.).

It should be appreciated by one of ordinary skill in the art that SNMP is not a protocol conventionally used to facilitate the distribution of applications across a distributed computing space. Instead, SNMP is well tailored for low-level (hardware) communications between network elements, which is how it is utilized in Barker's teachings. The low-level SNMP communications are conveyed directly from network elements to server, network elements (hubs, bridges etc.) communicating via SNMP are not disposed within software machines (like JAVA VIRTUAL MACHINES), but instead directly attached at a low level.

In contrast, application management is a higher-level (application level) communication that occurs between application components. This higher level communication can involve significant overhead (handled by the master agent and the mini-agent) necessary for an application component to utilize application resources. No equivalent structures, architectures,

or concepts are taught or contemplated by Barker. That is, Barker solves a different and unrelated problem (remotely managing network elements) than that solved by the Applicants. Barker is in a non-analogous field of endeavor from the Applicants claimed and disclosed invention.

Referring to independent claims 1, 10, 16, 24, 33, and 39, the Applicants explicitly claim the following:

(1) an **applicant manager** in a **first application host** that is a **software machine** configured to interpret compiled machine-independent code
(2) a **master agent** in a second **application host** that is a **software machine**
(3) a plurality of mini-agents, each in an application **host** (different from the first and second hosts) that is **a software machine**
(4) the application manager conveying commands to the master agent (that requires at least one resource), the master agent conveying a corresponding command to the mini-agent, the mini agent performing an operation responsive to the received command, the operation (utilizing the resource).

Barker is cited as teaching an application manager in a first application host, a master agent in a second application host, and a mini-agent in a third application host. Barker provides no such teachings. Instead, Barker teaches a network manager, a SNMP agent, and a SNMP agent, as shown by FIGS 3 and 4. Barker fails to teach or suggest an **application host**. Barker also fails to teach or suggest a **master agent** or a **mini-agent** as claimed by the Applicants.

Applicants are confused as to the exact elements that the Examiner believes to be equivalent to the elements of the disclosed invention. Specifically, the Examiner recurrently cites lines 24-34 of column 1 and FIG. 3 for teaching: the application hosts, application manager, the master agent, and the mini-agents. This passage indicates that a network element linked to a server can be managed from a client. This passage fails to mention master agents, mini-agents, or any system capable of managing distributed applications.

Presumably, the Examiner is stating the element management system client 28 is equivalent to the **first application host**, the element management system server 32 is equivalent to the **second application host**, and the application processor 80 (within the network element) or perhaps the network element itself? is equivalent to the **remote application hosts**.

Applicants teach that the first application host includes an application manager that initiates management commands to perform at least one management operation directed to at least one management resource. Presumably, the application manager is the represented by FIGS. 10-13.

Applicants also teach that the second application host includes a master agent. Applicants assume that the Examiner intends the object server 66 to be equivalent to the master agent. The object server 66 is generically defined at column 5, lines 35-40 as being a "single Unix process that provides most of the element management system server functionality." This assumption is made since this is the only illustrated component of the element management server 32 linked to the HTTP Web server component 58 and the HPOV processes 70, which provide a communication link to the element management system client and the application processor respectively.

Applicants further teach that the remote application hosts, each include a mini-agent. Applicants assume that the SNMP agent 81 is supposed to be equivalent to the mini agent.

Based upon these assumptions, Applicants conclude that Barker fails to contemplate an application host. As noted on page 12, lines 23-27, an application host can be an operating system session, a virtual machine or any other suitable process address space in which programs

can execute. Further, the claims define an application host to be a software machine configured to interpret compiled machine independent code.

The Examiner references column 1, lines 24-35 and FIG. 3 as teaching applications hosts. Presumably, the Examiner is claiming that items 32, 28, and 80 are "application hosts". Instead, each of the items 32, 28, and 80 are different types of hardware (namely a server, client, and a network element), none of which are defined as (or defined as including) software machines for interpreting code. Definitions in the specification (and in FIG. 2) detail the structure intended for items 32, 28, and 80, none of which include, imply, or infer a software machine.

The element management system server 32 is defined in FIG. 2 as including items 46, 48, 52, 55, 47, 54, 56, and 49, each typical of a robust hardware server hosting many Web applications. The components of the element management system server 32 is exhaustively detailed starting at column 8, line 1 ending at column 26 line 45. The detailed components are not ones conventionally present in a software machine. One of ordinary skill would not equate the element management system server 32 with a software machine or with an application host as defined and claimed by the Applicants.

The element management system client 28 is defined in FIG. 2 as including a Web browser 45 with many included JAVA applications 44. While these JAVA applications may very well utilize a software machine (like a JVM) to execute, the Web browser 45 as illustrated would not. Further, the element management system client 28 is listed in FIG. 1A as including a management computer 26. The client interface of the management compute 26 supports " Microsoft Internet Explorer and Netscape browsers as well as Web-enabled devices for PCs and

X-Terminals (as noted at column 4, lines 18-26). It is evident that the element management system client 28 is different from the application host defined and claimed by the Applicants.

The network element 14 is a hardware device (like a hub) for processing event and alarm notifications to the element management system via SNMP (as noted at column 4, lines 56-59). Such devices are not software machines or application hosts as defined and claimed by the Applicants.

Barker fails to contemplate a master agent. A master agent, as noted at page 13 lines 9-17, can respond to application manager commands without requiring the application managers to have specific knowledge of the particular application hosts in which the manageable resources reside.

In contrast, the object server 66 includes elements shown in FIG. 4, that permit the object server 66 to handle predetermined events, such as handling alarms via the alarm manager 120. The object server 66 is designed to accept specific client requests directed towards network elements at KNOWN network locations. Otherwise system administrators using the client 28 to manage network elements, cannot troubleshoot/ properly maintain the network elements when an alarm sounds. The object server 66 does not perform the same function (or any remotely similar function to) the master agent, as claimed by the Applicants.

Barker fails to contemplate a mini-agent. A mini-agent 24 includes a registry of manageable resources 26 that are exposed to the master agent 22. The mini-agent 24 is a local bifurcation of an agent layer as noted at page 13, lines 9-17, which is responsible for handling all activities pertaining to managed resources. As such, the mini-agent can register capabilities of managed resources that can be remotely accessed by the master agent 22.

In contrast, an SNMP agent is a management entity consisting of hardware and embedded software which responds to SNMP requests over Ethernet from an SNMP manager. SNMP agents and SNMP managers share a database of information, called the Management Information Based (MIB). Capabilities of the SNMP agent and MIB are defined by the Request for Comments (RFC) 1157. RFC 1157 is a standard that gathers statistical data about network traffic and the behavior of network components. RFC 1157 does not specify that SNMP agents are to register exposed capabilities. Further RFC 1157 defines SNMP as a protocol for managing networked devices designed as a standardized message conveyance, device querying protocol. RFC 1157 has nothing to do with managing applications deployed within a distributed computing space or with linking resources to these distributed applications. Further, one skilled in the art would not turn to RFC 1157 or its enhancements for teachings pertaining to deploying applications within a distributed computing space.

The Examiner cites column 4, lines 56-59 as teaching a registry and a command execution scheme. The cited lines merely reference the MIB of the SNMP protocol (defined by RFC 1157), which the Examiner has misinterpreted as being equivalent to a registry (exposing executable methods) of the mini-agent. By definition (RFC 1157) the SNMP agent 81 CANNOT be functional equivalent to the mini-agent 24.

Instead, the MID is a database of network management information that is used and maintained by a network management protocol such as SNMP or CMIP. The value of a MIB object can be changed or retrieved using SNMP or CMIP commands, usually through a GUI network management system. MIB objects are organized in a tree structure that includes public (standard) and private (proprietary) branches. Again, the MIB does not expose executable

methods to a master agent, as does the registry associated with a mini-agent as defined in the Applicants' specification.

Orenshteyn fails to cure the deficiencies of Barker. Orenshteyn fails to teach or suggest a master agent and/or a mini-agent. Instead, Orenshteyn teaches a means to retrieve an application from a remote location (securely) and to execute the retried application. That is, Orenshteyn teaches a remote server authentication methodology. Orenshteyn has nothing to do with remotely distributing applications or with remotely managing a network.

Not only does Orenshteyn fail to cure the deficiencies of Barker, but it is improper to combine teachings of Orenshteyn with teachings of Barker in the manner suggested by the Examiner. Orenshteyn is cited (page 2, paragraph 17) for teaching that an application host can be a software machine. The cited paragraph details that programs can be executed upon a client machine by first downloading an executable file, then executing the download file upon a client machine. The downloaded file can be a JAVA file interpreted by a JAVA interpreter. This teaching has nothing to do with network traffic, network elements, or SNMP.

The network element 14 of Barker is generally a fixed hardware device, like a hub. Barker never suggests that the network element 14 is to remotely execute remote programs, which seems a bizarre thing for a network element to do. To add this capability, a network element would have to be extended to include a updatable memory (most likely a persistent memory like a hard drive) upon which programs can be downloaded/executed and a software interpreter. This update would be costly and result in a potentially large exploitable security flaw. Neither Barker, Orenshteyn, or combinations thereof teach or suggest such a modification.

Moreover, Orenshteyn actually provides no teachings relevant to SNMP or to managing network devices, in general.

In paragraph 4 of the Office Action, claims 5-9, 11, 13-15, 20-22, 28-32, 34, 36-38, and 43-45 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over Barker in view of Orenshteyn and in further view of JAVA2. JAVA2 fails to cure the deficiencies of Barker and Orenshteyn. That is, JAVA2 fails to teach or suggest a application host, a master agent and/or a mini-agent, as claimed by the Applicants. Instead, JAVA2 is a generic White paper concerning JMX, which the Applicants do not claim to have invented.

It is not proper to combine the JAVA2 reference and the Barker reference for purposes of rejecting the Applicants claims, as neither reference teach or suggest the combinations asserted by the Examiner as obvious. Further, the suggested combination would fail to further the purpose of the Barker reference, which is to remotely administrate network elements.

The Examiner cites page 6 of JAVA2 as the motivation to combine references. The a paragraph does mention that protocol adaptors can be used to let external applications access a JMX agent and the MBeans contained within the agent. Barker, however, fails to teach or suggest using JMX agents or MBeans. There is no reason that MBeans would be located within the network elements used within Barker.

Applicants believe the Examiner is confused by the term agent and has inadvertently assumed an SNMP agent (defined by RFC 1157) is the same as a JMX agent (defined by the JMX architecture). Combining the references as suggested would require the use of BOTH an SNMP agent AND a JMX agent, as the two are fundamentally dissimilar and server different purposes.

Consequently, one of ordinary skill in the art familiar with JXM (and the material within the White Paper) would not contemplate using MBeans within a network element 14 described in Barker. Barker is not attempting to redefine network elements, which do not generally include MBeans. Instead Barker teaches using a client interface to remotely manage network devices. Unless MBeans were commonly contained within hubs, bridges, and other network elements (which they were and are still not) one would never contemplate adding a JMX agent to reference these beans (that would not exist). Certainly, Barker offers no teachings in this regard.

Referring to claim 5, Applicants claim that a master agent includes a JMX communication connector for communicating with the application manager and the mini-agents. Barker teaches that the object server (held to be equivalent to the master agent) is to communicate to the element management system client (held to be equivalent to the application manner) using a HTTP Web server. Barker further teaches that the object server is to communicate to the network element (held to be equivalent to the mini-agent) through HPOV processes 70). Applicants do not know where in the object server the Examiner proposes to add a JMX communication connector or how such a connector would affect the HTTP Web Server 58 and the HPOV processes 70. Regardless, Barker fails to teach or suggest such an arrangement.

Referring to claim 6, Applicants claim adding a JMX communication protocol adaptor to the master agent to provide a protocol adapted view to the application manager. This makes no sense in terms of the Barker disclosure. Why would it be desirable to add a JMX protocol adaptor to the Object Server 66 to provide a protocol adapted view to the network interface of FIGS. 10-13? Barker suggests no such arrangement, nor does Orenshteyn, nor does JAVA2.

Referring to claim 7, Barker does not teach or suggest that the Object Server is to remotely invoke any method. Such attempts would interfere with the SNMP mediator's 160 attempts to configure and poll requests/responses. It would not be advantageous for the element management server 32 to use RMI to invoke methods within network devices that it controls. The network management server 32 is instead directly linked to managed network elements using a low-level protocol, like SNMP. Accordingly, Barker fails to teach or suggest the use of RMI between the server and the network element.

Referring to claim 8, Barker fails to teach or contemplate network elements that include MBeans. Even if network elements internally included MBeans (for some unknown reason – MBeans result in substantial overhead not advantageous to network elements like hubs, bridges, routers) Applicants do not believe that it would be a good idea (nor obvious to a software developer) to provide an interface that would allow the MBeans to be manipulated. Such an interface would permit people having access to the network element to meddle with the MBeans, which would represent a significant network weakness or exploitable flaw, without resulting in any discernable benefit. Barker fails to teach or suggest network elements including MBeans, nor does Orenshteyn, nor does JAVA2. Neither to these references teach that an interface should be provided to manipulate MBeans within a network element (which does not typically include any MBeans).

Further, claims 12, 23, 35, and 46 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over under Barker in view of Orenshteyn in further view of JAVA2, in further view of Kukura. Kukura fails to cure the deficiencies of Barker, Orenshteyn, and JAVA2. That is, Kukura fails to teach or suggest a application host, a master agent and/or a mini-agent, as

claimed by the Applicants. Instead, Kukura is referenced because it uses RMI. Kukura is not related to Barker in any fashion, and there is no motivation within the references to form the combination suggested by the Examiner.

Instead, as mentioned above in reference to claim 7, Applicants know of no reason to use RMI to communicate between the network server of Barker and the network elements of Barker. Such a use seems to be counterintuitive to the purposes served by Barker. Regardless, neither Barker, Orenshteyn, JAVA2, nor Kukura provide any rational as to why a RMI should be used between a network server and network elements. Hence, no proper motivation exists to combine the references.

As noted above, neither Barker, Orenshteyn, JAVA2, Kukura, nor any combinations therein teach or suggest an application host, an application manager, a master agent, or a mini-agent. Further, neither Barker, Orenshteyn, JAVA2, Kukura, nor any combinations therein teach or suggest an architecture that facilitates the use of remotely located resources by distributed applications, as taught and claimed by the inventors. Additionally, no motivation exists to combine the references Barker, Orenshteyn, JAVA2, and Kukura for purposes of 35 U.S.C. § 103(a). Consequently, the 35 U.S.C. § 103(a) rejections to claims 1-47 should be withdrawn, which action is respectfully requested.

Applicants believe that this application is now in full condition for allowance, which action is respectfully requested. Applicants request that the Examiner call the undersigned if clarification is needed on any matter within this Amendment, or if the Examiner believes a telephone interview would expedite the prosecution of the subject application to completion.

Date: 9 Sep 2004

Respectfully submitted,

Gregory A. Nelson, Registration No. 30,577
Richard A. Hinson, Registration No. 47,652
Brian K. Buchheit, Registration No. 52,667
AKERMAN SENTERFITT
Post Office Box 3188
West Palm Beach, FL 33402-3188
Telephone: (561) 653-5000